ROLL-A-BALL

Chapter 1. The Ball and the Roll

1.- Abrimos Unity Hub. Creamos un nuevo proyecto. Como nombre del proyecto pondremos “Roll a Ball”. Seleccionamos como localización la carpeta “Unity” que hemos creado en Mis Documentos. Seleccionamos la plantilla 3D.

2.- Comprobamos en la vista de proyecto la existencia de SampleScene dentro de la carpeta Scenes. Ejecutaremos File > Save con frecuencia a lo largo del tutorial.

3.- Creamos el campo de juego como un plano. Nombramos el plano como “Ground” y reseteamos su posición. Cambiamos la escala del plano en x = 2 y z = 2.

4.- Creamos una esfera y la nombramos como “Player”. Reseteamos su posición y la situamos en y = 0.5. Asignamos a la esfera un componente Rigidbody.

5.- Seleccionamos el objeto Player en la jerarquía. Creamos un nuevo Script desde “Add component” del inspector de objetos. Seleccionamos New Script y damos como nombre “PlayerController” y de tipo CSharp.

6.- Creamos una nueva carpeta en Assets llamada Scripts. Una vez creado lo arrastramos de la raíz de la vista de proyectos a la carpeta de Scripts. Hacemos doble click en el script para editarlo:

```
private Rigidbody rb;
public float speed;

void Start () {

    rb = GetComponent<Rigidbody>();
}

void FixedUpdate () {

    float x = Input.GetAxis("Horizontal");
    float z = Input.GetAxis("Vertical");
    Vector3 movement = new Vector3(x, 0.0f, z);
    rb.AddForce(movement * speed);

}
```

7.- En el inspector de propiedades de Player asignamos un valor a Speed = 1.

8.- Pulsamos Play y probamos con las teclas cursoras. Podemos experimentar con diferentes valores de Speed (5, 10, 20).

Chapter 2. The Third-Person View

9.- Ubicamos mejor la cámara. La subimos (posición) a y=10 y la rotamos a x=45.

10.- Arrastramos la cámara sobre el jugador en la vista de jerarquía. Ahora, si con una vista general en la escena (2 by 3 Layout), pulsamos Play y movemos el jugador, vemos como se mueve la cámara vinculada rotando. (Ya que está como hija del jugador).

11.- Quitar la cámara de hija del Player arrastrándola fuera de él. Vamos a asociarla al jugador a través de un script. Creamos un script para la cámara llamado CameraController.

```
public GameObject player;
private Vector3 offset;

void Start () {
    offset = transform.position;
}

void LateUpdate () {
    transform.position = player.transform.position + offset;
}
```

12.- Es necesario arrastar el Player desde la vista de jerarquía a la nueva propiedad de la cámara llamada Player. A continuación, pulsamos Play y probamos el movimiento de la cámara siguiendo a la bola.

Chapter 3. Beyond the walls

13.- Creamos un Empty Game Object, le llamamos “Walls” y reseteamos su posición al origen.

14.- Creamos un nuevo cubo y le llamamos “West Wall” y lo arrastramos sobre “Walls” para hacerlo su descendiente. Cambiamos la escala del cubo a x=0.5, y=2, z=20.5 y lo arrastramos hasta situarlo en posición x=-10 (o introducimos el valor manualmente).

15.- Seleccionamos West Wall en la vista de jerarquía y lo duplicamos. Llamamos al nuevo cubo “East Wall” y lo colocamos en posición x=10.

16.- Duplicamos East Wall y lo llamamos “North Wall”. Podemos rotar el muro y = 90 o bien aplicar una nueva escala al cubo x=20.5, y=2, z=0.5 y establecer su posición como x = 0 y z=10.

17.- Duplicamos North Wall y lo llamamos South Wall con una posición en z = -10.

18.- Pulsamos Play y probamos cómo funcionan los muros.

Chapter 4. The Top-Down Camera

19.- Creamos una nueva cámara y la llamamos “TopDown Camera”. Reseteamos su posición. Seleccionamos la vista Game para ver como se ve la cámara.

20.- Elevamos la posición de la cámara a y = 20 y la rotamos en x = 90 para que mire hacia abajo. Cambiamos la proyección a ortográfica y el tamaño (“Size”) a 14.

21.- Reducimos la ventana de la cámara “ViewPort Rect” situándola en x = 0.7 y = 0.7 w = 0.28 H = 0.28. Cambiamos “Clear Flags” a “Depth only”.

Chapter 5. Heads-Up Display

22.- Creamos un nuevo objeto de tipo UI > Text. Cambiamos su nombre a “Position Text”. Cambiamos su propiedad Text en el inspector a “Position Text”. Su color a un color que se vea bien en contraste con la escena. Y su “Font Size” a 20.

23.- En el componente “Rect Transform” del texto pulsamos en el icono de alineación y en el desplegable que aparece. Con las teclas SHIFT y CTRL pulsadas hacemos click en la casilla intersección de “top” y “left”. Con esto situamos el texto arriba a la izquierda.

24.- Para separarlo un poco de los márgenes ponemos en “Pos X” 10 y en “Pos Y “ -10.

25.- Añadimos al script “PlayerController” lo siguiente:

Al inicio del script:

```
using UnityEngine.UI;
```

Dentro de la clase Player Controller:

```
public Text positionText;
```

y dentro de la función FixedUpdate:

```
positionText.text = string.Format(
    "x = {0:f2}\ny = {1:f2}\nVx = {2:f2}\nVy = {3:f2}",
     transform.position.x, transform.position.z,
     rb.velocity.x, rb.velocity.z);
```

26.- Arrastramos el objeto “Position Text” desde la jerarquía a la nueva propiedad “Position Text” que aparece en el inspector del objeto “Player”.

Chapter 6. Twelve cubes

27.- Creamos un nuevo cubo y lo llamamos PickUp reseteando su transformada al origen. Centramos la vista de la escena en él.

28.- Seleccionamos al Player y des chequeamos su nombre en el inspector para que no interfiera en la visión del nuevo cubo creado.

29.- Situamos el cubo en y=0.5, lo hacemos más pequeño para que parezca que flota reescalándolo a x=0.5, y=0.5 y z=0.5. Y lo rotamos 45 grados en cada eje para que llame más la atención.

30.- Crear un script para el cubo llamado “Rotator”. Ver ayuda sobre la componente “Transform” en [http://docs.unity3d.com/ScriptReference/Transform.html]. Examinar los métodos Translate y Rotate. Escribir:

```
void Update () {
    transform.Rotate(new Vector3(15, 30, 45) * Time.deltaTime);
}
```

31.- Pulsar Play y comprobar el giro del cubo.

32.- Crear en la vista de proyectos una carpeta para guardar Prefabricados “Prefabs”. Arrastrar el objeto PickUp desde la jerarquía a la nueva carpeta de prefabricados en la vista de proyectos.

33.- Crear un nuevo Empty Game Object llamado PickUps. Resetear su posición al origen. Arrastrar el objeto Pickup al nuevo PickUps.

34.- En la escena pulsar en el eje “y” del manejador de ejes para obtener una vista desde arriba. Hacer zoom para ver todo el campo de juego. Pasar de “Local” mode a “Global” mode para poder mover el objeto paralelo al plano de juego. (Se puede probar que pasa en “Local” mode y luego hacer un “undo”).

35.- Con la opción de duplicar (CTRL-D) crear 12 objetos PickUp y situarlos formando un círculo. Pulsar “Play” y comprobar que todos rotan.

Chapter 7. The collision

36.- Volver a activar el objeto “Player”. En su componente Sphere Collider pulsar el símbolo de ayuda para acceder a la ayuda relativa al componente. Ver la opción en la ayuda de “Switch to Scripting” que nos permite acceder a la ayuda para codificar el componente. Ver Collider.OnTriggerEnter. Nos permite detectar el contacto entre dos objetos.

37.- Buscar en la documentación (scripting) de Unity “GameObject” y localizar su variable “tag” y su función “SetActive” para comprender su funcionalidad.

38.- En el script PlayerController añadimos el siguiente método:

```
void OnTriggerEnter(Collider other) {
    if (other.gameObject.tag == "PickUp") {
        other.gameObject.SetActive(false);
    }
}
```

39.- Seleccionamos el prefabricado PickUp en el visor de proyecto y vamos a la propiedad “Tag” en el inspector de propiedades. Seleccionamos “Add Tag…”. Pulsamos el “+” para añadir y escribimos “PickUp”. Pulsamos Save. Volvemos a seleccionar el prefabricado PickUp en el visor de proyectos y en el inspector seleccionamos el tag “PickUp”.

40.- Seleccionamos el prefabricado PickUp y en su componente Box Collider marcamos “Is Trigger” para que las colisiones las gestione nuestro script. Probamos el funcionamiento pulsando Play y moviendo el Player hacia los cubos.

41.- Asignamos al prefabricado PickUp un Rigidbody para que sea más eficiente ya que Unity así lo considera como collider dinámico. Probamos de nuevo y comprobaremos que los cubos ahora caen.

42.- Para que los cubos no caigan al dar Play (caen porque con el Rigidbody están sometidos a gravedad y el colisionador lo estamos tratando por código), podemos desactivar la gravedad en el prefabricado PickUp o aún mejor, marcar los cubos como “Is Kinematic” en su componente Rigidbody para que no se vean sometidos a las fuerzas en el motor de física.

Chapter 8. The final countdown (\m/>_<\m/)

43.- En el script “PlayerController” añadir:

```
private int count;
```

En el método “Start” añadir:

```
count = 0;
```

Después de “other.gameObject.SetActive(false);” añadir:

```
count = count + 1;
```

44.- Creamos un nuevo objeto de tipo UI > Text y lo llamamos “Count Text”. Lo situamos en la posición “top”, “center” y establecemos unos márgenes de Pos X = 20 y Pos Y = -10. Como tamaño de letra seleccionamos 20. Asignamos un color adecuado.

45.- Añadir el siguiente código a “PlayerController”:

```
public Text countText;

void SetCountText() {

    countText.text = "Count: " + count.ToString();
}
```

En el método “Start” añadir:

```
SetCountText();
```

Añadirlo también en “OnTriggerEnter”, a continuación de count = count + 1:

46.- Arrastrar el nuevo objeto de texto sobre el inspector de propiedades de Player, donde ha aparecido una nueva propiedad llamada “Count Text”. Pulsar Play y probar el funcionamiento.

47.- Creamos un nuevo objeto de tipo UI > Text y lo llamamos “Win Text”. Lo situamos en la posición “middle”, “center” y establecemos su “Font Size” a 60, Pos X = 200, Width = 668 y Height = 100.

48.- En PlayerController añadir lo siguiente:

```
public Text winText;
```

En el método “Start” añadir:

```
winText.text = "";
```

En el método “SetCountText” añadir:

```
if (count >= 12) {
    winText.text = "YOU WIN !!";
}
```

49.- Arrastrar el nuevo objeto “Win Text” sobre la nueva propiedad “Win Text” que aparece en el inspector de propiedades del objeto “Player”. Guardar la escena con File/Save scene. Pulsar Play y probar.

50.- Pulsamos File/Build Settings… Pulsamos “Add Open Scenes”. Pulsamos el botón “Build” y seleccionamos una nueva carpeta que creamos en el directorio raíz (En el mismo que está contenida la carpeta Assets) llamado “Builds”. Como nombre del ejecutable escribimos “Roll-a-Ball_v01” y pulsamos Guardar.

The End